UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/712,463	11/12/2003	Judith Schwabe	P-4181CIP	9131

24209 7590 09/24/2007
GUNNISON MCKAY & HODGSON, LLP
1900 GARDEN ROAD
SUITE 220
MONTEREY, CA 93940

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
09/24/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/712,463	SCHWABE ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *24 August 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-78* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-78* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 8/24/07.

As indicated in Applicant's response, claims 1, 4, 11, 13, 20, 23, 30,32, 39, 42, 49, 51, 58, 61, 68, 70, 77-78 have been amended. Claims 1-78 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 18, 37, 56, 75 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 12, 25, 38, 51 of U.S. Patent No. 7,107,581 (hereinafter '581). Following are examples as to how the claims are conflicting.

As per instant claim 18, '581 claim 12 also recites converting first instruction to a second instruction based on relationship between base of operand, the base of the second instruction smaller than that of the first instruction, converting instructions in a chain such that operands with potential overflows potential beyond the base of the second base, i.e. matching by

changing type of operand of the second instruction to match with that of the first instruction base, the chain bounded by the second instruction and a third instruction being the source of the operand being subjected to the modification (see '581 claim 1); determining potential overflow associated with said second instruction and generating an output stack ('581 claim 11); indicating said 2nd instruction has potential overflow if it does not equal said first type of operand, if it does not remove such overflow, if it creates said overflow (see '581 claim 12, 1st para); indicating said 2nd instruction has potential overflow if it does not equal said first type of operand, if it does not remove such overflow, if it does not create said overflow (see '581 2nd para, claim 12: ...*if overflow is not possible*...), if said 2nd instruction propagates said overflow, and if at least one operand is one input stack has potential overflow (*).

Although '581 claim 12 does not recite 'indicating said second instruction overflow if said second type does not equal said first type ... indicating said second ... if said second instruction does not create potential overflow ... if ... propagates potential overflow ... at least one input stack has potential overflow' (as recited in instant claim 17), claim 18 and claim 17, together, can be addressed together with the subject matter of '581 claim 12. That is, the matter recited in instant claim 17 amounts to a slight variation of --yet equivalent to-- the subject matter of instant claim 18; hence would be treated as subsumed under claim 18 and therefore obvious when treating it with the '581 claim 12, as set forth above (see *).

'581 claim 12 does not recite '*one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input*

instruction'; but in view of '581 claim 12 reciting of 'determining potential overflow associated with said second instruction and generating an output stack based on execution of said 2nd instruction (see '581 claim 11) ... and indicating said 2nd instruction... has potential overflow, and if at least one operand ... one input stack has potential overflow (see '581 claim 12), the above limitation is considered analogously disclosed because output stack with operand being modified entails input stack operands potential overflow problem, according to the above from '581 to accomplish the above conversion operating on the operands associated with an input stack.

Nor does '581 recite '*changing the type of instructions ... to equal said 2nd type if said operand type is less than said second type*'; but '581 claim 1 recites that said second base (of a second processor) smaller than that of said first base so that the bounding of instructions between said 2nd instruction and a 3rd instruction -being a source of a potential overflow- is such that this chain involves converting to a wide base if the target operand (of the second instruction) carries a potential overflow that is beyond that of the second instruction. Hence, the above limitation is considered analogously disclosed by virtue of equivalent teaching albeit the apparent different wording in the language of the claims.

Instant claims 37, 56, and 75 recite the same limitations as claim 18 and '581 claims 25, 38, 51 recite the same subject matter of '581 claim 12; hence instant claims 37, 56, and 75 are also conflicting with the subject matter claimed in '581 claims 12, 38, 51 by virtue of the rationale as set forth above.

4. Claims 16, 35, 54, 73 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 53, 54 of U.S. Patent No. 7,107,581.

Further, **instant claims 16, 35, 54, 73** conflict with the subject matter of '581 claims 53, 54 by virtue of the input stack analysis above, i.e. output stack generating based on potential overflow of operand of 2nd instruction reads on input stack associated with 1st instruction for which a 2nd instruction of operand type belonging to lower base is to matched and converted to prevent overflow.

5. Claims 18, 37, 56, 75 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 12, 25, 38, 51 of U.S. Patent No. 7,207,037 (hereinafter '037). Following are examples as to how the claims are conflicting.

By virtue of '037 claims 12, 25, 38, 51 being substantially similar to claims 12, 25, 38, 51 of U.S. Patent No. 7,107,581 instant claims 18, 37, 56, 75 for being non-patentably distinct from those claims in '781 will be deemed obvious over '037 claims 12, 25, 38, 51 for the same reasons.

6. Claims 16, 35, 54, 73 are rejected on the ground of non-statutory obviousness-type double patenting as being unpatentable over claims 53, 54 of U.S. Patent No. 7,207,037.

By virtue of '037 claims 53, 54 being substantially similar to claims 53, 54 of U.S. Patent No. 7,107,581 instant claims 16, 35, 54, 73 would be rejected as obvious variants of '037 claims 53, 54 as set forth above.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-78 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yellin et al., USPN: 5740441 (hereinafter Yellin), and further in view of Wilkinson et al., USPN: 6,308,317 (hereinafter Wilkinson).

As per claim 1, Yellin discloses a method for arithmetic expression (e.g. col. 4, lines 42-51; *integer add* - col. 9, lines 42-52; col. 17, TABLE 1: *isub, idiv, imul, iadd*) optimization, comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction (e.g. Fig. 4B-D; col. 6, lines 6-38);

reconfigure said first instruction to operate as one operand of a second type, said second type smaller than said first type (e.g. *overflow* – Fig. 4C, 4D; *updated...modified ... by the current instructions* - col. 10, line 58 to col. 11, line 6; *and improving execution time efficiency* – col. 5, line 65 to col. 6, line 4), said reconfiguring based at least in part on the relative size of said first type and said second type (Note: overflow analysis for each successor instruction reads on size of destination place in stack for a successor operand being smaller to take required size of upper instruction in the data flow – see Fig. 4A-B; *different data types* -- col. 21, lines 16-38); and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of

instructions (e.g. Fig. 4, 5 – Note: any process that iterates one instruction at a time until all are processed reads on chain of instructions bounded by a start and a current element being processed), said chain bounded by said second instruction and a third instruction that is the source of said at least one operand (see col. 19-21: Do until there are no instructions whose changed bit is set ... Do for each successor instruction ...Merge the Virtual stack ... Verification Success – Note: algorithm to merge – see col. 11-12).

But Yellin does not explicitly that the above reconfiguring includes *converting said first instruction to a second instruction different from the first instruction* and operable *with a one operand of a smaller type than said first type*; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type *if said operand type is less than said second type*. However, Yellin discloses adding data and rearranging stack calling structure relative to discrepancies of first and second type (see col. 1, lines 60-67; col. 20, lines 24-35; col. 21, lines 12-37; step 394, Fig. 4B) hence has suggested modifying the runtime instruction as verified by the stack to accommodate data being ported from a larger size operand --or higher platform-- to a smaller size operand -- or lower platform (see col. 5, lines 14-64; *multiple computer platforms, underlying instruction sets* - col. 1, lines 7-14). Analogous to Yellin's approach as to modifying stack runtime Java bytecodes in order to accommodate an executing platform receiving/using data coming from a platform with higher architecture base (see Yellin: Background of the invention, col. 1), Wilkinson discloses modifying stack operands from a larger base platform to operands of lesser size that would fit the target platform (see Fig. 10-11; col. 11, lines 4-48), hence disclose a form of conversion from a word operand to a byte size operand of a lesser size than the word-based

type operand. Based on the common endeavor by Yellin or Wilkinson to alleviate extraneous security and runtime resources of a given lower platform (e.g. receiving a JVM application destined for larger microcomputer into integrated circuit's microcontroller in portable devices) when loading operands for each instruction in Java method prior to runtime, it would have been obvious for one skill in the art at the time the invention was made to provide the operand type replacement as approached by Wilkinson so to support Yellin's loading process by way of converting at the receiving platform a larger type operand to a smaller size operand. One would be motivated to do so because providing operands of smaller size and familiar to the (smaller target platform) runtime as replacement for those corresponding operands of a higher size base as purported by Wilkinson and via Yellin's attempt to obviate overflow would alleviate Yellin's application resources for dealing with exception due to overflow, obviate the resources of the JVM interpreter role as desired by Yellin (see by Yellin: col. 15, lines 12-30), and according to Wilkinson, the stack space of the smaller platform (e.g. Fig. 21-24) might be loaded with data compliant to the platform thus expediting code execution via prechecking and reshuffling of bytecode, operands (see Fig. 9-11), enhancing the runtime with a safer method supporting a device whose resources are not equipped for checking large data being provided (see Wilkinson: BACKGROUND, col 3).

As per claims 2-3, see Yellin as said first instruction is arithmetic (col. 4, lines 42-51; *integer add* - col. 9, lines 42-52; col. 17, TABLE 1: *isub, idiv, imul, iadd*); a non-arithmetic, type-sensitive instruction (see col. 16-18: Table 1).

As per claims 4-5, see Yellin (in view of Wilkinson's conversion) for repeating said validating, said converting and said matching for instructions that comprise a program (see Figs 4-5) and linking each instruction to input instructions in all control paths (step 366 – Fig. 4A).

As per claim 6, see Yellin (col. 5, lines 14-64) wherein said first instruction is defined for a first processor having a first base; and said second instruction is defined for a second processor having a second base.

As per claims 7-8, Yellin wherein said first processor comprises a Java Virtual Machine (see Fig. 1-2); and in view of the obviousness rationale of claim 1 see Wilkinson (e.g. Fig. 1-2) wherein said second processor comprises a Java Card Virtual Machine with resource-restraint platform, i.e. operable with lower operand type; according to which rationale, Yellin combined with Wilkinson(see Wilkinson Fig. 11) discloses wherein said first processor comprises a 32-bit processor; and said second processor comprises a resource-constrained 16-bit processor.

As per claims 9-10, Yellin discloses said at least one input stack comprises a plurality of input stacks, said plurality of input stacks further comprising a first input stack and a second input stack; and said validating further comprises comparing operand types of corresponding entries in said first input stack and said second input stack (refer to claim 1 in light of Figs. 4); wherein said comparing further comprises indicating an error if the types of at least said first at least one stack entry and said second at least one stack entry are not equivalent (Fig. 4D, 4E, 4F).

As per claim 11, Yellin by virtue of the overflow analysis and size mismatch (refer to claim 1) and the conversion from a larger size to a smaller size in claim 1, would have rendered (based on the teaching of Wilkison) obvious the following:

setting said second type to a smallest type;

setting said second type to the type of an operand in said at least one input stack if said smallest type is less than said type of said operand (see rationale of obviousness using teaching of Wilkinson); and setting said second type to a type that is larger than said smallest type if said smallest type is greater than said type of said operand, if said operand has potential overflow, if said second instruction is sensitive to overflow and if said second type is less than said first type (see rationale of claim 1; according to which any type if predicted to overflow beyond a smaller size will be replaced, i.e. overflow analysis as by Yellin inherently teaching setting a smallest type to a larger type when said smallest type is itself greater than type of any operand potentially risking being overflowed).

As per claim 12, Yellin (in view of Wilkinson) discloses wherein said smallest type is the smallest type supported by a target processor (see rationale of claims 7-8).

As per claims 13-14, Yellin (in view of Wilkinson) discloses wherein said smallest type is the smallest type determined during a previous pass of said converting (Yellin: Figs 4-5);

wherein said third instruction is not a source instruction (any instruction being loaded does not have to be a source for a operand that is predicted to overflow); and

said changing further comprises: recursively examining input instructions until said third instruction is obtained; and setting the type of said third instruction to equal said second type (Note: in view of the recursive process to verify by Yellin and changing to a smaller operand size of the target platform by Wilkinson, any replacement to match the platform operand size being included in the verification process – see Yellin: Fig. 4-5 – will read no setting a third instruction to be equal to a second instruction previously replaced in the chain)

As per claim 15, Yellin (in view of Wilkinson) discloses (see Verification algorithm: col. 19-21) wherein said third instruction comprises a source instruction (Note: within a chain of instructions source instruction is the identified instruction starting from which replacing a original first operand with target platform operand of a lesser size begins); and said

changing further comprises: recursively examining input instructions until said third instruction is obtained; setting the type of said third instruction to equal said second type; and repeating said changing for each input instruction of said third instruction (see rationale of claim 14).

As per claim 16, Yellin discloses comprising recording, said recording comprising: determining potential overflow associated with said second instruction (see col. 9, lines 30-60); but Yellin does not disclose generating an output stack based at least in part on execution of said second instruction. But Wilkinson discloses replacement of operands for a stack (see Fig. 11, 16, 18) to support Yellin's verification endeavor as set forth in claim 1; hence the stack being modified to include (as an output stack) adjusted operands as per the combination set forth in claim 1 would also be an obvious limitation to enable Yellin to provide a smaller platform to make efficient use of its runtime environment, utilizing Wilkinson's approach for the reasons as set forth above in claim 1.

As per claim 17, Yellin discloses (combined with Wilkinson) wherein said determining further comprises: indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if said second instruction creates potential overflow; and indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction

does not remove potential overflow, if said second instruction does not create potential overflow, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow (see Yellin: col. 9, lines 30 to col. 10, line 34 – Note: determining a need for change in a chain of instructions wherein operands of smaller size would be loaded in stack to *replace* – as set forth in claim 1 -- larger size operands deemed not suitable for the target platform reads on if said second instruction *does not remove* overflow or *might not necessarily create overflow*, and the automated process by which a chain in which all instructions are recursively verified reads on propagating the replacement due to potential overflow).

As per claim 18, Yellin discloses (combined with Wilkinson)wherein said determining further comprises: indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if overflow is possible based at least in part on the type of said second instruction and the relationship between said first type and said second type; and indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, if overflow is not possible based at least in part on the type of said second instruction and the relationship between said first type and said second type, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow (see Yellin in view of Wilkinson – as set forth in claim 1 -- according to the rationale of claim 17).

As per claim 19, Yellin alone does not disclose creating a new output stack based at least in part on one of said at least one input stack; updating said new output stack based at least in

part on operation of said second instruction; and indicating another instruction conversion pass is required if said new stack does not equal a previous output stack. But in view of the automated and iterative verification process by Yellin combined with the operands replacement by Wilkinson as set forth in claim 1, the constant updating of stack per iteration pass in regard to what is deemed potentially conflicting in regard to stack overflow (see Yellin: Figs 4-5), it would have been obvious for one skill in the art to make many passes to update the runtime stack according to Yellin's approach using marking of unmatched operand allocation, so that combining with Wilkinson's replacement of larger size operands with more compliant size operands, Yellin's verification would be ready for execution with a stack that is repeatedly updated, and this would further alleviate runtime resources of small platforms using smaller operand type as set forth in claim 1.

As per claim 20, Yellin discloses a method for arithmetic expression optimization, comprising: step for validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

step for reconfiguring said first instruction to operate on at least one operand of a second type, said second type smaller than said first type, said converting based at least in part on the relative size of said first type and said second type; and

step for matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching including a chain

being bounded; said chain bounded by said second instruction and a third instruction that is the source of said at least one operand;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly that the above reconfiguring includes converting said first instruction to a second instruction different from the first instruction and operable with a one operand of a smaller type than said first type; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. Such limitation has been addressed in claim 1 using Wilkinson.

As per claims 21-38, refer to rejections set forth to claims 2-19 respectively.

As per claim 39, Yellin discloses a program storage device readable by a machine, embodying a program of instructions executable by the machine to perform a method for arithmetic expression optimization, the method comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

reconfiguring said first instruction to operate on at least one operand of a second type, said second type smaller than said first type, said converting based at least in part on the relative size of said first type and said second type; and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of

instructions being bounded, said chain bounded by said second instruction and a third instruction that is the source of said at least one operand;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly that the above reconfiguring includes converting said first instruction to a second instruction different from the first instruction and operable with a one operand of a smaller type than said first type; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. But the limitation has been addressed in claim 1 using a combination with Wilkinson.

As per claims 40-57, refer to rejections set forth to claims 2-19 respectively.

As per claims 58-76, these are the apparatus version of claims 1-19; hence are rejected with the corresponding rejections as set forth therein, respectively.

As per claim 77, Yellin discloses a method of using an application software program including arithmetic expression optimization of at least one instruction targeted to a processor, the method comprising receiving the software program on said processor, said software program optimized according to a method comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

reconfiguring said first instruction to operate on at least one operand of a second type, said second type smaller than said first type, said converting based at least in part on the relative size of said first type and said second type; and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of instructions being bounded, said chain bounded by said second instruction and a third instruction that is the source of said at least one operand; and executing said at least one instruction on said processor;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly that the above reconfiguring includes converting said first instruction to a second instruction different from the first instruction and operable with a one operand of a smaller type than said first type; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. But the limitation has been addressed in claim 1.

As per claim 78, Yellin discloses a controller configured to execute a virtual machine, the virtual machine capable of executing a software application comprising a plurality of previously optimized instructions, the instructions optimized by a method comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

reconfiguring said first instruction to operate on at least one operand of a second type, said second type smaller than said first type, said converting based at least in part on the relative size of said first type and said second type; and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of instructions being bounded, said chain bounded by said second instruction and a third instruction that is the source of said at least one operand;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly that the above reconfiguring includes converting said first instruction to a second instruction different from the first instruction and operable with a one operand of a smaller type than said first type; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. But the limitation has been addressed in claim 1.

Nor does Yellin disclose that the controller is a smart card having a microcontroller embedded therein; but in view of the rationale as set forth in claim including Wilkinson's Java card machine, this smart card controller limitation would also have been obvious because of the benefits as set forth in that rationale.

Response to Arguments

9. Applicant's arguments filed 8/24/07 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

Double Patenting Rejection:

(A) Applicants have submitted that claim 1, 16, 18 include four operations and should be considered as a whole to establish a proper obviousness rejection (Appl. Rmrks pg. 26). The argument does not map each of the cited portions (conflicting language) proffered in the Office Action to the claim language being rejected, and for each portion, correspondingly identify the possible distinguishing aspects thereof respective recited limitations being addressed. That is, Applicants have submitted that claim 53 of '581 must suggest each limitation of claim 16; and that the Rejection fails to provide such suggestion. The level of one of ordinary skill in the art has it that what appears to not explicit can be interpreted as being strongly implied; and such implied teaching can serve a fuel as to a suggestion leading to more possibilities based on the existence of the more explicit teachings in the claims. The rejection has carefully put forth a derivation in terms of some level of knowledge based on the scenario of stack analysis and operand mapping in the context of overflow prevention; i.e. such knowledge being the suggestion of a need to accommodating the size conflict with proper conversion or reconfiguring of stack storage; such that this leads to converting into a second form of smaller instruction (*). Notwithstanding that, Applicants fail to identify each part of the claim as mentioned above, and correspondingly, point out where the discrepancies in teaching lie.

The language of the claims leads to interpretation and such interpretation has been based upon for the Office to map the corresponding prior art's teachings. If the purpose is to clearly point out where the cited parts fail to disclose or suggest the elements being recited, Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. The rejection has shown what is

equivalent in terms of claimed steps, what is not; and based on suggested teaching and level of one of ordinary skill in the art, the motivation to combine for an obvious feature has been properly laid out (refer to *). What is being conveyed in the above argument amounts to a general allegation that there is a morphing of language by the Office to leap into what is recited as four distinct steps; however, this type of allegation only signifies that a prima facie case is not present: Applicants omit to steadily dissect limitation per limitation and provide in a convincing manner any distinguishing or non-obvious details thereof respective to the corresponding claimed counterparts of the conflicting Application.

35 USC § 103 Rejection:

(B) Applicants have submitted that each reference should support the interpretation that forms the basis of the rejection of claims 1-78; and when Yellin steps simply abort the verification process, there is no suggestion to make one instruction to obtain another instruction (Appl. Rmrks pg 27, para 2-4). The reason for Java bytecodes to be verified at runtime with heavy emphasis on stack resources, secure and safe Java thread allocation, garbage collection and mapping of correct object dereferencing or type resolution has been a known concept and a industrial desirable commodity to enable Java to be ported in applications among heterogeneous platforms without conflicts that would otherwise damage the environment of the receiving platforms as contemplated in both Yellin and Wilkinson. For one of ordinary skill in the art of using JVM and bytecodes across machines (see Yellin: multiple computer platforms, underlying instruction sets - col. 1, lines 7-14), particularly when the platforms to be ported into are resources-restraint devices with smaller micro-architecture not suitable for a larger Java code originated from a higher architecture, Yellin's approach in detecting stack overflow in light of

the above size discrepancy strongly suggests a need to accommodate a runtime stack of a platform recipient of the downloaded bytecodes by way of a form of conversion to overcome size differential as mentioned above. The Office Action has pointed out how the converting step would be obvious in view of such overflow threat, thread safe checking and portable applications in terms of transmitted bytecodes across machines. It is worth mentioning that when a given architecture (smaller size platform) receives a bytecode with larger size operands, the only possible way to overcome this would be to convert such larger bytecode operand so that it fit the resident architecture of receiving environment; and the resulting operand (from this converting) would have to be different from the original incompatible bytecode/operand, lest no execution at the receiving platform can be achieved; i.e. first instruction converted into a second instruction. Applicants fail to point out the reason as to why Yellin would not be willing to convert a higher bit word into a smaller byte operand when faced with such platform architectural differential. Further, an obviousness rationale operates from the point where some feature or limitation is determined as not being reasonably conveyed by a main reference; and using the secondary reference, combines the teachings or suggestion of both references in order to rationalize a combinational type of teachings by virtue of which the whole claimed invention (parts covered by the main reference and parts that are deemed obvious) would be fulfilled and so carried out with reasonable level of success. Applicants fail to point out how such combined teachings would be yielding inapposite result or would not be fit a reasonable combination in the first place. The argument is not persuasive.

(C) Applicants have submitted that for claim 1, Yellin must suggest starting with a first instruction and ending with a second instruction after a conversion based on relative size of said

first and second instruction (Appl. Rmrks pg. 28,top, middle) and that the second reference fails to provide the remedy for what Yellin fails to teach. The explanation as to render obvious such converting step based on relative size of a original bytecode and a targeted runtime bytecode has been largely set forth in section B above, with emphasis on how the suggestion requirement has been derived. The argument is therefore not sufficient to overcome the rejection.

(D) The rest of the argument amounts to relying on the points made against Yellin in the earlier claims; hence would be falling under the ambit of the response set forth in section B above.

In all, the claims will stand rejected as set forth in the Office Action.

Conclusion

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tuan A Vu
Patent Examiner,
Art Unit 2193
September 17, 2007